ELSEVIER The Journal of Logic Programming 38 (1999) 371–386

THE JOURNAL OF LOGIC PROGRAMMING

Technical Note

Sharing is optimal

Agostino Cortesi *, Gilberto Filé

Universita Ca Foscari, Computer Science Department, Via Torino 155, 30170 Mestre-Venezia, Italy

Received 13 January 1998; received in revised form 4 May 1998; accepted 17 August 1998

Abstract

One of the most popular abstract domains used for the analysis of logic programs is the domain `Sharing` which expresses the fact that computed substitutions bind variables to terms containing common variables. Despite the fact that this domain is widely used and studied, it is not yet known whether its abstract operations are complete or at least optimal. We solve this open question showing that the operations of *lub* and *projection* of `Sharing` are complete (and thus optimal), whereas that of *unification* is optimal, but not complete.

Keywords: Static analysis; Abstract interpretation

1. Introduction

Abstract interpretation [1] is a general framework for defining and relating different semantics of transition systems. The most popular application of abstract interpretation is the design of data flow analysis for programming languages. Many data flow analyses have been designed for logic programs and often these analyses have the goal of inferring variable sharing information. This information is useful for many purposes: AND-parallelism [2], freeness [3], program transformation [4,5]. Probably the most popular domain for inferring sharing information is `Sharing` [6]. Despite the number of applications of `Sharing` and of studies of its properties [7–10], a fundamental property of `Sharing` is not yet known: whether its operations are complete [1,11] or at least optimal [1].

We answer this question here showing that the operations *lub* and *projection* of `Sharing` are complete (and thus optimal), whereas the *unification* operation is only optimal and not complete. To this end we also show the correctness of all three operations of `Sharing`. As a matter of fact, the correctness of the *unification* of `Sharing` has been already shown in Refs. [6,10], but in these works the domain is different from that considered here and thus we give the full proof of this result

* Corresponding author. Tel.: +39 41 290 8428; fax; +39 41 290 8419; e-mail: cortesi@dsi.unive.it.

0743-1066/99/$ – see front matter © 1999 Elsevier Science Inc. All rights reserved.
PII: S0743-1066(98)10026-2

too. It is worth mentioning that, in order to give a precise foundation to our results, we consider here an important point that arises when dealing with the concrete unification and that is often neglected: do our results depend on a particular concrete unification algorithm? We show, exploiting a result of [12], that as long as idempotent most general unifiers (*mgu*'s) are considered, the sharing does not depend on which *mgu* is considered.

The paper is organized as follows. Section 2 recalls some preliminary definitions. In Sections 3 and 4 the concrete interpretation `Rsub` and the abstract interpretation `Sharing` are introduced, each with its three operations: *lub*, *projection* and *unification*. In Section 5 we show that *lub* and *projection* of `Sharing` are complete and in Section 6 we show that abstract unification is optimal (but not complete). Finally, Section 7 discusses recent related works, in particular [9,10], and concludes.

2. Preliminaries

2.1. Substitutions

Let $\mathbf{V}$ be a countable set of variables, $FP(\mathbf{V})$ be the set of finite subsets of variables of $\mathbf{V}$, and $\mathbf{T}_{\mathrm{V,G}}$ be the set of finite terms over $\mathbf{V}$ and an alphabet of function symbols $\mathbf{G}$. A *substitution* σ is a function in $\mathbf{V} \to \mathbf{T}_{\mathrm{V,G}}$ such that $\sigma(x) \neq x$ only for a finite number of variables x. The *set of support* and the variable range of σ are given by $supp(\sigma) = \{x \mid \sigma(x) \neq x\}$ and $var\text{-}range(\sigma) = \bigcup\{Var(\sigma x) \mid x \in supp(\sigma)\}$, where $Var(t)$ denotes the set of variables occurring in t. The set of variables occurring in σ is $Var(\sigma) = supp(\sigma) \cup var\text{-}range(\sigma)$. A substitution may be specified by listing its non-trivial bindings, viz., $\sigma = \{x/\sigma(x) \mid x \in supp(\sigma)\}$. Given two substitutions σ_1 and σ_2, their *composition*, denoted $\sigma_2 \circ \sigma_1$, is $\{x/\sigma_2(\sigma_1(x)) \mid x \in supp(\sigma_1) \text{ and } \sigma_2(\sigma_1(x)) \neq x\} \cup \{y/\sigma_2(y) \mid y \in supp(\sigma_2) \setminus supp(\sigma_1)\}$. A substitution σ is *idempotent* when $\sigma \circ \sigma = \sigma$. It is well-known that σ is idempotent iff $supp(\sigma) \cap var\text{-}range(\sigma) = \emptyset$.

If there exists ϑ such that $\sigma_2 = \vartheta \circ \sigma_1$, then σ_1 is *more general* than σ_2. Let $E = \{t_1 = s_1, \ldots, t_k = s_k\}$ be a set of term equations. If σ makes $\sigma(t_i)$ syntactically identical to $\sigma(s_i)$ for each $(t_i = s_i) \in E$, σ is called a *unifier* of E.

A *most general unifier* of E, *mgu*(E), is an idempotent unifier of E that is more general than all other unifiers of E. The set E is in *solved form* if it has the form $\{x_1 = t_1, \ldots, x_n = t_n\}$ where each x_i is a distinct variable occurring in none of the terms t_j. In this case, the substitution $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ is an idempotent *mgu* of E. Any idempotent substitution σ is an *mgu* of the corresponding set of equations in solved form, denote by $Eq(\sigma)$.

We write *Subst* for the set of idempotent substitutions. Although *Subst* is not closed under composition, in a step of the execution of a logic program in which $\vartheta \circ \sigma$ is constructed, it is always the case that, $var\text{-}range(\vartheta) \cap supp(\sigma) = \emptyset$, which, provided that ϑ and σ are idempotent, ensures that $\vartheta \circ \sigma$ is also idempotent.

2.2. Abstract interpretations

According to [1], a data-flow analysis for a programming language L is a non-standard (abstract) semantics of L. Both the standard and an abstract semantics of L are obtained interpreting a generic semantics, where such an interpretation consists of a domain of data-descriptions and of some operations on these data-descriptions.

Let us recall some basic definitions [1,13]. Let C and D be complete lattices. Two functions $\gamma_{DC} : D \to C$ and $\alpha_{CD} : C \to D$ form a *Galois connection* between D and C if

$$\forall c \in C \text{ and } \forall d \in D : \alpha_{CD}(c) \sqsubseteq_D d \Longleftrightarrow c \sqsubseteq_C \gamma_{DC}(d).$$

The function γ_{DC} is called *concretization* and α_{DC} is called *abstraction*. They are said to be *adjoint* because one is determined by the other [13]. For instance, $\forall d \in D, \gamma_{DC}(d) = \sqcup_C\{c \mid c \in C, \alpha_D(c) \sqsubseteq_D d\}$. The dual relation holds for α_{DC}. This definition is equivalent to requiring that concretization and abstraction are monotonic and that the following two conditions hold: $\forall c \in C, \gamma_{DC}(\alpha_{CD}(c)) \sqsupseteq_C c$ and $\forall d \in D, \alpha_{CD}(\gamma_{DC}(d)) \sqsubseteq_D d$.

A Galois connection is said to be a *Galois insertion* when γ_{DC} is injective or, equivalently, when α_{CD} is onto. In this case, $\forall d \in D$, $\alpha_{CD}(\gamma_{DC}(d)) = d$.

Assume that an abstract operation μ corresponds to a concrete one *op*. We say that μ is *correct* with respect to *op* when

$$\forall d \in D, \mu(d) \sqsupseteq_D \alpha_{CD}(op(\gamma_{DC}(d))).$$

The operation μ is *optimal* when the above disequation becomes an equality. In [1,11], the stronger notion of *completeness* of an abstract operation is introduced and discussed. The abstract operation μ is *complete* when

$$\forall c \in C, \alpha_{CD}(op(c)) = \mu(\alpha_{CD}(c)).$$

Obviously, an optimal abstract operation is also correct. When there is a Galois insertion, it is also easy to see that a complete operation is also optimal, but the converse is not true.

3. The concrete interpretation $\langle \mathsf{Rsub}, \sqcup_{Rs}, \pi_{Rs}, U_{Rs}\rangle$

Before introducing the concrete interpretation, some explanations are due. In this paper we adopt the classical abstract interpretation approach of [1,13]. In this approach, both the concrete and the abstract semantics are obtained by instantiating a common generic semantics. This approach has the advantage of allowing modular correctness proofs: instead of proving the correctness (optimality, completeness) of the whole abstract semantics w.r.t. the concrete one (for instance the SLD semantics), one can show the correctness (optimality, completeness) of each basic operation.

This approach has two important consequences on the concrete interpretation [14] adopted here. The elements of this domain consist of pairs where the first component is a set of substitutions (as usual) and the second component is a finite set of variables that specifies the *variables of interest*, i.e., the variables the analysis wants to deals with. This second component is needed in order to account, in the concrete domain, for the fact that the abstract semantics computes values concerning only finite sets of variables (in general, the variables of the clauses of the analyzed program).

The second consequence is that the concrete interpretation includes also a projection operation. This operation is the concrete counterpart of the abstract projection which, on the contrary, has a very important role in making the abstract semantics finitely computable.

Although non-standard, this choice of concrete domain allows to use an uniform notation for concrete/abstract values and operations, and this is the reason we adopt it [14,7,8]. Moreover, it enjoys two positive features: its values are almost identical to the sets of substitutions computed by Prolog programs, and the projection operation is very simple, matching the feeling that projection does not belong to the concrete semantics, but it is "inherited" from the abstract semantics, where it is necessary for achieving finiteness, as said above.

3.1. Domain

The set $\texttt{Rsub} = [\wp(\mathit{Subst}) \times \mathit{FP}(\mathbf{V})] \cup \{\top_{Rs}, \bot_{Rs}\}$. Rsub stands for restricted substitutions. The partial order of Rsub is defined as follows. $\top_{Rs}$ is the largest element, $\bot_{Rs}$ is the smallest one, whereas $[\Sigma_1, U_1] \sqsubseteq_{Rs} [\Sigma_2, U_2]$ iff $U_1 = U_2$ and $\Sigma_1 \subseteq \Sigma_2$.

3.2. Least upper bound and greatest lower bound

The operation $\sqcup_{Rs}$, which produces the least upper bound of two elements of Rsub, is as follows: for any $k \in \texttt{Rsub}$, $\top_{Rs} \sqcup_{Rs} k = \top_{Rs}$, $\bot_{Rs} \sqcup_{Rs} k = k$, whereas

$$[\Sigma_1, U_1] \sqcup_{Rs} [\Sigma_2, U_2] = \begin{cases} [\Sigma_1 \cup \Sigma_2, U_1] & \text{if } U_1 = U_2, \\ \top_{Rs} & \text{otherwise.} \end{cases}$$

The greatest lower bound is defined on non-trivial elements by $[\Sigma_1, U] \sqcap_{Rs} [\Sigma_2, U] = [\Sigma_1 \cap \Sigma_2, U]$. In the other cases, it is $\bot_{Rs}$. Rsub is a complete lattice w.r.t. $\sqsubseteq_{Rs}$.

3.3. Projection

The concrete projection π_{Rs} is the identity on the first argument when this is either $\top_{Rs}$ or $\bot_{Rs}$, otherwise it is

$$\begin{aligned} \pi_{Rs} : \ & \texttt{Rsub} \times \mathit{FP}(\mathbf{V}) \longrightarrow \texttt{Rsub} \\ & ([\Sigma, U_1], U_2) \longmapsto [\Sigma, U_1 \cap U_2]. \end{aligned}$$

3.4. Unification

A unique operation [14] is considered that accounts for both *forward* and *backward* unifications. Its first argument is the set of substitutions computed at calling time, the second one is the set of substitution computed when returning from the call, and the third one is a (idempotent) ***mgu*** of the two atoms that are unified through the call. For using this operation as forward unification it suffices to set the second argument to the identity substitution. In order to define the concrete unification $\mathbf{U}_{Rs}$, it is convenient to introduce first the following function $\mathbf{u}_{Rs}$:

$$\begin{aligned} \mathbf{u}_{Rs} : \ & \mathit{Subst} \times \mathit{Subst} \times \mathit{Subst} \longrightarrow \mathit{Subst}, \\ & (\sigma_1, \sigma_2, \delta) \longmapsto \boldsymbol{mgu}(Eq(\sigma_1) \cup Eq(\sigma_2) \cup Eq(\delta)). \end{aligned}$$

$\mathbf{U}_{Rs}$ is strict: if either of the first two arguments is $\bot_{Rs}$, the result is $\bot_{Rs}$. Otherwise, if one of these is $\top_{Rs}$, the result is $\top_{Rs}$. The other cases are as follows: assume that

$Var(\delta) \subseteq U_1 \cup U_2$; then

$$\begin{aligned}&\mathbf{U}_{Rs} : \mathtt{Rsub} \times \mathtt{Rsub} \times \mathit{Subst} \rightarrow \mathtt{Rsub},\\ &\quad([\Sigma_1, U_1],\ [\Sigma_2, U_2], \delta) \mapsto [\{\mathbf{u}_{Rs}(\sigma_1, \sigma_2, \delta) \mid \text{where } \sigma_1 \in \Sigma_1, \sigma_2 \in \Sigma_2,\\ &\qquad\qquad \text{and } Var(\sigma_1) \cap Var(\sigma_2) = \emptyset\},\ U_1 \cup U_2].\end{aligned}$$

Notice that σ_1 and σ_2 are required to be renamed apart. Intuitively, this works because σ_1 deals with the variables of the calling clause whereas σ_2 with those of the called clause only.

4. The abstract interpretation $\langle \mathtt{Sharing}, \sqcup_{Sh}, \pi_{Sh}, \mathbf{U}_{Sh} \rangle$

4.1. Domain

The abstract domain Sharing has been proposed by Jacobs and Langen in Ref. [6] in order to represent variable aliasing, covering, and groundness. Here we present a slightly different domain wrt the original proposal in that our domain shows explicitly the variables of interest [7,8]. This is necessary in order to use this domain in a real static analysis in which values on different sets of variables (for instance, the variables of the clauses of the analyzed program) are generally computed.

$$\mathtt{Sharing} = \{[A, U] \mid A \subseteq \wp(U), (A \neq \emptyset \Rightarrow \emptyset \in A), U \in FP(\mathbf{V})\} \cup \{\top_{Sh}, \bot_{Sh}\}.$$

Sharing is partially ordered: $\top_{Sh}$ is the largest element and $\bot_S$ is the smallest one; for the other elements, $[A_1, U_1]$, $\sqsubseteq_{Sh}$ $[A_2, U_2]$ iff $U_1 = U_2$ and $A_1 \subseteq A_2$.

4.2. Least upper bound and greatest lower bound

The least upper bound of any two elements $[A_1, U_1]$ and $[A_2, U_2]$ is defined by

$$[A_1, U_1] \sqcup_{Sh} [A_2, U_2] = \begin{cases} [A_1 \cup A_2, U_1] & \text{if } U_1 = U_2,\\ \top_{Sh} & \text{otherwise.}\end{cases}$$

The greatest lower bound, if $U_1 = U_2 = U$ is $[A_1 \cap A_2, U]$. Otherwise, it is $\bot_{Sh}$. The domain Sharing is a complete lattice w.r.t. $\sqsubseteq_{Sh}$.

4.3. Abstraction

Let for any $y \in \mathbf{V}, occ(\sigma, y) = \{z \in \mathbf{V} \mid y \in Var(\sigma(z))\}$. The abstraction function between Sharing and Rsub maps $\bot_{Rs}$ to $\bot_{Sh}$ and $\top_{Rs}$ to $\top_{Sh}$, whereas it is defined on non-trivial elements by

$$\alpha_{Rs\,Sh}(c) = \begin{cases} \text{if } c = [\{\sigma\}, U] & \text{then } [\{occ(\sigma, y) \cap U \mid y \in \mathbf{V}\}, U],\\ \text{if } c = [\Sigma, U] & \text{then } \sqcup_{Sh} \{\alpha_{Rs\,Sh}([\{\sigma\}, U]) \mid \sigma \in \Sigma\}.\end{cases}$$

Intuitively, an element of the first component of $\alpha_{Rs\,Sh}([\{\sigma\}, U])$ is a set of variables in U that under σ share the same variable. Observe that a variable of U is ground in $[A, U]$ iff it does not appear in any set of the abstract state. Clearly, $\alpha_{Rs\,Sh}$ is monotonic. In order to show that it admits an adjoint function and that the two functions form a Galois connection, it suffices by Ref. [13] to show the following result.

Proposition 4.1. *The function $\alpha_{Rs\,Sh}$ is join-complete, i.e.,*

$$\forall X \subseteq \texttt{Rsub},\ \alpha_{Rs\,Sh}(\sqcup_{Rs}X) = \sqcup_{Sh}\{\alpha_{Rs\,Sh}(x) \mid x \in X\}.$$

Proof. The limit cases in which X is empty or it contains $\top_{Rs}$, or it consists of $\perp_{Rs}$ only, are immediate. Then, assume that X contains some pairs $[\Sigma, U]$. All these pairs must have equal second component, otherwise both sides of the equation are trivially $\top_{Rs}$. Then, it is sufficient to remind the definition of $\alpha_{Rs\,Sh}$ to conclude. □

The adjoint function of $\alpha_{Rs\,Sh}$ is:

$$\gamma_{Sh\,Rs}([A, U]) = \sqcup_{Rs}\{[\Sigma, U] \in \texttt{Rsub} \mid \alpha_{Rs\,Sh}([\Sigma, U]) \sqsubseteq_{Sh} [A, U]\}.$$

Actually, $\alpha_{Rs\,Sh}$ and $\gamma_{Sh\,Rs}$ define a Galois insertion between Rsub and Sharing. In fact the following Proposition shows that $\alpha_{Rs\,Sh}$ is onto.

Proposition 4.2. *For any $[A, U] \in$ Sharing, there exists $\sigma \in Subst$ such that*

$$\alpha_{Rs,Sh}([\{\sigma\}, U]) = [A, U].$$

Proof. Let $A = \{B_1, \ldots B_m\}$. Consider $y_1, \ldots, y_m \in \mathbf{V} \setminus U$. For each $x \in U$, let $I_x = \{i \mid 1 \leqslant i \leqslant m, x \in B_i\}$. The substitution σ is defined by:

$$\sigma(x) = \begin{cases} f(y_{j_1}, \ldots, y_{j_h}) & \text{if } I_x = \{j_1, \ldots, j_h\}, \\ a & \text{if } I_x = \emptyset. \end{cases}$$

It is easy to check that for such a σ it holds $\alpha_{Rs,Sh}([\{\sigma\}, U]) = [A, U]$. □

Example 4.1. Let us illustrate the proposition above by a simple example. Let $[A, U]$ be an element of Sharing with $U = \{x_1, x_2, x_3, x_4\}$, and $A = \{\emptyset, \{x_1, x_2, x_3\}, \{x_2, x_3\}, \{x_1, x_3\}, \{x_3\}\}$. The substitution σ obtained according to Proposition 4.2 is based on the following correspondence between elements in A and variables in $\mathbf{V} \setminus U$.

$$\{\underbrace{x_1, x_2, x_3}_{y_1}\}, \{\underbrace{x_2, x_3}_{y_2}\}, \{\underbrace{x_1, x_3}_{y_3}\}, \{\underbrace{\ x_3\ }_{y_4}\}.$$

The resulting substitution is

$$\sigma = \{x_1/f(y_1, y_3), x_2/f(y_1, y_2), x_3/f(y_1, y_2, y_3, y_4), x_4/a\}.$$

4.4. Projection

In Sharing Projection is the identity on $\perp_{Sh}$ and $\top_{Sh}$, whereas in the other cases it is defined through set-intersection:

$$\begin{aligned} \pi_{Sh} : \texttt{Sharing} \times FP(\mathbf{V}) &\to \texttt{Sharing} \\ ([A_1, U_1], U_2) &\mapsto [\{B \cap U_2 \mid B \in A_1\}, U_1 \cap U_2] \end{aligned}$$

4.5. Unification

Some auxiliary functions are useful [6] in order to define the abstract unification function $\mathbf{U}_{Sh}$:

- The *closure under union* of $A \in \wp(\wp(\mathbf{V}))$, denoted A^*, is the smallest superset of A satisfying $X \in A^* \wedge Y \in A^* \to (X \cup Y) \in A^*$.

- The component of $A \in \wp(\wp(\mathbf{V}))$ that is *relevant* to a term t, denoted $rel(A,t)$, is the set $\{S \in A \mid Var(t) \cap S \neq \emptyset\}$.
- If $A, A' \in \wp(\wp(\mathbf{V}))$, the *cross product* $A \otimes A'$ is $\{(S \cup S') \mid S \in A^*, S' \in A'^*\}$.
- The basic unification step is performed by

$$\mathbf{u}_{Sh} : \wp(\wp(\mathbf{V})) \times Subst \longrightarrow \wp(\wp(\mathbf{V}))$$

$$\forall A_0 \in \wp(\wp(\mathbf{V})),\ \forall \delta \in Subst, \delta = \{x_1/t_1, \ldots, x_m/t_m\}$$
$$\mathbf{u}_{Sh}(A_0, \delta) = \mathbf{amgu.args}([x_1, \ldots, x_m], [t_1, \ldots, t_m], A_0)$$

$$\mathbf{amgu.args}([\,], [\,], B) = B$$
$$\mathbf{amgu.args}([x_1|\bar{x}], [t_1|\bar{t}], B) = \mathbf{amgu.args}(\bar{x}, \bar{t}, \mathbf{amgu}(x_1, t_1, B))$$
$$\mathbf{amgu}(x, t, B) = (B - (rel(x,B) \cup rel(t,B))) \cup (rel(x,B) \otimes rel(t,B))$$

- The backward/forward unification $\mathbf{U}_{Sh}$ is defined as follows. Let $[A, U], [A', U'] \in$ `Sharing`, with $U \cap U' = \emptyset$, and let $\delta \in Subst$ with $Var(\delta) \subseteq U \cup U'$.

$$\mathbf{U}_{Sh} : \mathtt{Sharing} \times \mathtt{Sharing} \times Subst \longrightarrow \mathtt{Sharing}$$

$$\mathbf{U}_{Sh}([A, U], [A', U'], \delta) = [\mathbf{u}_{Sh}(A \cup A', \delta), U \cup U'].$$

5. Lub and projection of `sharing` are complete

Correctness of least upper bound and projection operations have been proven, in a different setting, in Ref. [10] (see Section 7). Here, we show a stronger result: that these operations are complete too. Completeness of lub operation follows immediately from Proposition 4.1.

Theorem 5.1. *The lub of* `Sharing` *is complete.*

Theorem 5.2. *π_{Sh} is complete with respect to π_{Rs}, i.e., $\forall c \in$* `Rsub` *and $U_2 \in FP(\mathbf{V})$, $\alpha_{Rs\,Sh}(\pi_{Rs}(c, U_2)) = \pi_{Sh}(\alpha_{Rs\,Sh}(c), U_2)$.*

Proof. If c is either $\top_{Rs}$ or $\bot_{Rs}$, the result follows from the definition. Otherwise, consider $c = [\Sigma, U_1]$. We get

$$\begin{aligned}
\alpha_{Rs\,Sh}(\pi_{Rs}([\Sigma, U_1], U_2)) &= \alpha_{Rs\,Sh}([\Sigma, U_1 \cap U_2]) \\
&\qquad \text{by definition of } \pi_{Rs} \\
&= [\{occ(\sigma, y) \cap (U_1 \cap U_2) \mid y \in \mathbf{V},\ \sigma \in \Sigma\}, U_1 \cap U_2] \\
&\qquad \text{by definition of } \alpha_{Rs\,Sh} \\
&= \pi_{Sh}([\{occ(\sigma, y) \cap U_1 \mid y \in \mathbf{V},\ \sigma \in \Sigma\}, U_2] \\
&\qquad \text{by definition of } \pi_{Sh} \\
&= \pi_{Sh}(\alpha_{Rs\,Sh}([\Sigma, U_1]), U_2) \\
&\qquad \text{by definition of } \alpha_{Rs\,Sh}.
\end{aligned}$$

6. Unification of `Sharing` is optimal

This Section is organized as follows. First we show that, when computing a *mgu* of a set of equations E, all idempotent *mgu*'s of E carry the same sharing information. Thus the actual unification algorithm used has no effect on the sharing information as long as idempotent *mgu*'s are computed. After this, we prepare the background for the optimality proof. First some notation is fixed and some technical facts are proven. Then, it is shown that $\mathbf{U}_{Sh}$ is correct and optimal w.r.t. $\mathbf{U}_{Rs}$, but not complete.

6.1. Idempotent mgu's and sharing

In Section 3 of Ref. [12] the following result is shown:

Theorem 6.1. *Two idempotent mgu's μ_1 and μ_2 of some set of equations E satisfy the following two conditions:*
1. $Var(\mu_1) = Var(\mu_2)$;
2. *There is* $\{x_1/y_1, \ldots, x_n/y_n\} \subseteq \mu_1$, *with all the* y_i *distinct, such that* $\mu_2 = \{y_1/x_1, \ldots, y_n/x_n\} \circ \mu_1$.

From this result it is easy to show the following theorem.

Theorem 6.2. *Let μ_1 and μ_2 be two idempotent mgu's of the same equation set. For any* $U \in FP(V)$, $\alpha_{Rs\,Sh}([\{\mu_1\}, U]) = \alpha_{Rs\,Sh}([\{\mu_2\}, U])$.

Proof. When $U = Var(\mu_i)$ (recall that by point (1) of Theorem 6.1, $Var(\mu_1) = Var(\mu_2)$), the result follows immediately from Theorem 6.1, point (2). When $U \neq Var(\mu_i)$, just use the definition of $\alpha_{Rs\,Sh}$. □

6.2. Notation

In what follows, let $[A_1, U_1]$ and $[A_2, U_2] \in$ `Sharing` with $U_1 \cap U_2 = \emptyset$. Let also $R_0 = A_1 \cup A_2$ and $U_0 = U_1 \cup U_2$ and $\delta = \{x_1/t_1, \ldots, x_n/t_n\} \in Subst$ such that $Var(\delta) \subseteq U_0$. Recall that

$$\mathbf{U}_{Sh}([A_1, U_1], [A_2, U_2], \delta) = [\mathbf{u}_{Sh}(R_0, \delta), U_0].$$

The operation $\mathbf{u}_{Sh}(R_0, \delta)$ treats the bindings of δ one at the time. We denote by R_i the subset of $\wp(U_0)$ computed after having handled the first i bindings. The operation $\mathbf{u}_{Sh}$ generates sets of variables by taking the union of already present sets. Observing the computation of $\mathbf{u}_{Sh}(R_0, \delta)$ one can reconstruct which elements of R_0 are unioned in order to generate each element of R_n. [1] If an element of R_n is generated in several ways, just any one of them will be considered.

In what follows we will consider $S \in R_n$ and assume that it is generated by taking the union of $K = \{B_1, \ldots, B_k\} \subseteq R_0$. Clearly, if k is 1 then $S = B_1 \in R_0$.

[1] More precisely, one could mimic the computation of $\mathbf{u}_{Sh}(R_0, \delta)$ computing $\{B_1, \ldots, B_h\}$ whenever that function computes $\cup_{i\in[1,h]} B_i$.

6.3. Technical results

Assume the notation fixed above. K satisfies the following 3 facts:

Fact 1. $\forall i \in [1,n], x_i \in S \iff S \cap Var(t_i) \neq \emptyset$.

Fact 2. If $S \notin R_0$ then $\forall j \in [1,k], \exists i \in [1,n]$ such that $B_j \cap Var(x_i = t_i) \neq \emptyset$.

Fact 3. Let us define the following relation $\Re$ between elements of K: $B_i \Re B_j$ if and only if there exists $l \in [1,n]$, such that $B_i \cap Var(x_l = t_l) \neq \emptyset$ and $B_j \cap Var(x_l = t_l) \neq \emptyset$. Let $\Re^+$ be the transitive closure of $\Re$; then it must be that $\forall i,j \in [1,k], B_i \Re^+ B_j$.

Proof of Fact 1. ($\Rightarrow$) If $x_i \in S$ then x_i appears also in R_i, and then, by definition of $\mathbf{U}_{Sh}$, it must be that $\cup R_{i-1} \cap Var(t_i) \neq \emptyset$. Moreover, each element $B \in R_i$ that contains x_i, contains also a variable in $Var(t_i)$. Thus, this must be also true for S, that contains at least one of these sets. The same reasoning proves the ($\Leftarrow$)–direction. □

Proof of Fact 2. It suffices to observe that if $\exists j \in [1,k]$, such that $\forall i \in [1,n]$, $B_j \cap Var(x_i = t_i) = \emptyset$, then, $\mathbf{u}_{Sh}$ could not combine B_j with any other element of R_0, and thus S could not be produced. □

Proof of Fact 3. Assume that there are i and $j \in [1,k]$ such that $B_i \Re^+ B_j$ is false. In this case, K can be partitioned in two parts: $K' = \{B \in K \mid B_i \Re^+ B\}$ and $K'' = K - K'$, such that each binding of δ has variables in common with at most one of $\bigcup K'$ and $\bigcup K''$. From this, by induction on n, it follows that elements of K' and K'' are never unioned in the computation of $\mathbf{u}_{Sh}(R_0, \delta)$. Thus, the set S (see Section 6.2) cannot belong to $\mathbf{u}_{Sh}(R_0, \delta)$. Contradiction. □

6.4. Correctness

As already mentioned in the Introduction, the correctness of the abstract unification of `Sharing` was already shown in Ref. [6]. In a different setting, also Ref. [10] proves it. However, since the domain we consider is different and for the sake of completeness, we give the full proof of this result too.

Theorem 6.3. *The abstract unification $\mathbf{U}_{Sh}$ is correct with respect to $\mathbf{U}_{Rs}$.*

Proof. Let $[A_1, U_1], [A_2, U_2] \in$ `Sharing` with $U_1 \cap U_2 = \emptyset$. Let also $R_0 = A_1 \cup A_2$ and $U_0 = U_1 \cup U_2$ and $\delta = \{x_1/t_1, \ldots, x_n/t_n\} \in Subst$ such that $Var(\delta) \subseteq U_0$. The claim of the Theorem is:

$$\mathbf{U}_{Sh}([A_1, U_1], [A_2, U_2], \delta) \sqsupseteq_{Sh} \alpha_{Rs,Sh}(\mathbf{U}_{Rs}(\gamma_{Sh,Rs}([A_1, U_1]), \gamma_{Sh,Rs}([A_2, U_2]), \delta)).$$

Let for $i \in [1,2], \sigma_i \in \Sigma_i$, where $[\Sigma_i, U_i] = \gamma_{Sh,Rs}([A_i, U_i])$ such that $Var(\sigma_1) \cap Var(\sigma_2) = \emptyset$ as required by the concrete unification, cf. Section 3.4. Because of this assumption, an (idempotent) *mgu* of $Eq(\sigma_1) \cup Eq(\sigma_2)$ is simply the union of the bindings of these two substitutions, i.e., $\rho_0 = \sigma_1 \cup \sigma_2$. Therefore, it is true that

$$\begin{aligned}\mathbf{u}_{Rs}(\sigma_1, \sigma_2, \delta) &= mgu(Eq(\sigma_1) \cup Eq(\sigma_2) \cup Eq(\delta)) \\ &= mgu(Eq(\rho_0) \cup Eq(\delta)).\end{aligned}$$

It is convenient for this proof to consider the following construction of this *mgu*: let $Eq(\delta) = \{x_1 = t_1, \ldots, x_n = t_n\}$, first we compute $mgu(\rho_0(x_1 = t_1)) = \mu_1$, then if $\rho_1 = \mu_1 \circ \rho_0$, we compute $mgu(\rho_1(x_2 = t_2)) = \mu_2$, then we apply $\rho_2 = \mu_2 \circ \rho_1$ to $x_3 = t_3$ and so on. At the end of this process the obtained substitution ρ_n is an idempotent *mgu* of $E = Eq(\rho_0) \cup Eq(\delta)$ and thus, by Lemma 6.2, whatever we can prove about its sharing, this holds for any other idempotent *mgu* of E.

Below, the notation fixed in Section 6.2 will be used again. In particular, the computation of $\mathbf{u}_{Sh}([R_0, U_0], \delta])$ is performed in a way similar to that just described for $\rho_1, \ldots, \rho_n$: the bindings of δ are considered one at the time and R_i is the value obtained after having considered the first i bindings of δ.

We prove by induction that

$$\forall i \in [0, n],\ \ \gamma_{Sh\,Rs}([R_i, U_0]) \sqsupseteq_{Rs} [\{\rho_i\}, U_0].$$

From this fact the theorem follows by observing that $\mathbf{U}_{Sh}([A_1, U_1], [A_2, U_2], \delta) = [R_n, U_0]$, and by using the definition of Galois connection.

Base $(i = 0)$. From the assumption, $\forall i \in [1, 2], \sigma_i \in \gamma_{Sh\,Rs}([A_i, U_i])$, and $U_1 \cap U_2 = \emptyset$. It follows that $[\{\rho_0\}, U_0] \in \gamma_{Sh\,Rs}([R_0, U_0])$.

Step $(i > 0)$. Assume now that the above statement holds for $[R_{i-1}, U_0]$ and ρ_{i-1}, and consider $\mu_i = mgu(\rho_{i-1}(x_i = t_i))$ and $\rho_i = \mu_i \circ \rho_{i-1}$.

Observe that, by the definition of the operator *occ*, $occ(\rho_i, y) \cap U_0 \neq \emptyset$ only for $y \in var\text{-}range(\rho_i) \cup (U_0 \setminus supp(\rho_i))$. Thus, we need to show that for each y either in the *var-range* of ρ_i or in $U_0 \setminus supp(\rho_i)$, it is true that $(occ(\rho_i, y) \cap U_0) \in R_i$. We distinguish two cases.

1. $y \notin Var(\rho_{i-i}(x_i = t_i))$. This implies that

$$occ(\rho_{i-1}, y) \cap Var(x_i = t_i) = \emptyset. \quad (+)$$

On the one hand, from $(+)$ it follows that all the bindings of ρ_{i-1} concerning y are unchanged in ρ_i. Moreover, no new binding concerning y can be added in ρ_i because $y \notin Var(\rho_{i-1}(x_i = t_i))$. Thus, $occ(\rho_i, y) = occ(\rho_{i-1}, y)$.

On the other hand, $(+)$ implies that $(occ(\rho_{i-1}, y) \cap U_0) \notin rel(x_i = t_i, R_{i-1})$ and thus, by definition of $\mathbf{u}_{Sh}$, $occ(\rho_{i-1}, y) \cap U_0$ is not modified after precessing the ith binding of δ. Hence, $occ(\rho_i, y) \cap U_0 \in R_i$.

2. $y \in Var(\rho_{i-1}(x_i = t_i))$. Let $B = occ(\mu_i, y)$. Observe that $B \neq \emptyset$ since $\mu_i = mgu(\rho_{i-1}(x_i = t_i))$. In ρ_i, y will be shared by all the variables that in ρ_{i-1} share some variable in B. More precisely,

$$occ(\rho_i, y) \cap U_0 = \bigcup\{occ(\rho_{i-1}, w) \cap U_0 \mid w \in B\}.$$

By induction hypothesis, all the sets $occ(\rho_{i-1}, w)$, $w \in B$, are in R_{i-1}. In order to conclude the proof we show that $\mathbf{u}_{Sh}$ computes their union.

Since each $w \in B$ is in $Var(\rho_{i-1}(x_i = t_i))$, it must be that $occ(\rho_{i-1}, w) \cap Var(x_i = t_i) \neq \emptyset$. Thus, since $U_0 \supseteq Var(x_i = t_i)$, it follows that $occ(\rho_{i-1}, w) \cap U_0 \in rel(x_i = t_i, R_{i-1})$. This shows that all the sets $occ(\rho_{i-1}, w) \cap U_0$ that must be unioned to produce $occ(\rho_i, y) \cap U_0$, are considered by $\mathbf{u}_{Sh}$ when it handles $x_i = t_i$.

It remains to show that some of these sets are in $rel(x_i, R_{i-1})$ and some in $rel(t_i, R_{i-1})$. If this is the case, in fact, $\mathbf{u}_{Sh}$ produces, among others, also the set $occ(\rho_i, y) \cap U_0$.

To this end, observe that B contains at least one variable, call it z_1, that occurs in $\rho_{i-1}(x_i)$ and at least one, call it z_2, that occurs in $\rho_{i-1}(t_i)$, in fact, μ_i is a unifier of $\rho_{i-1}(x_i = t_i)$ and thus y must be on both sides of $\mu_i(\rho_{i-1}(x_i = t_i))$. Two possible cases may apply to z_1:

- $\rho_{i-1}(x_i) \equiv x_i \equiv z_1$. In this case, since $x_i \in U_0$, $x_i \in occ(\rho_{i-1}, x_i) \cap U_0 \neq \emptyset$. Hence $occ(\rho_{i-1}, x_i) \cap U_0$ will be in $rel(x_i, R_{i-1})$.
- $\rho_{i-1}(x_i) \not\equiv x_i$. In this case, $occ(\rho_{i-1}, z_1) \cap U_0$ contains x_i and thus this set is included in $rel(x_i, R_{i-1})$ too.

A similar reasoning applies to z_2. This concludes the proof. □

6.5. Optimality

Let us turn now to the proof of optimality of the abstract unification. The idea behind the proof is that for each set S in the result of an abstract unification $\mathbf{U}_{Sh}([A_1, U_1], [A_2, U_2], \delta)$ we may identify a pair of concrete substitutions σ_1, σ_2, represented by each of the two abstract states, such that the result of the concrete unification $u_{Rs}(\sigma_1, \sigma_2, \delta)$ is a substitution whose abstraction is $[\{S, \emptyset\}, U_1 \cup U_2]$. The substitutions σ_1 and σ_2 are defined according to the following idea. The set S is obtained as the union of sets $\{B_1, \ldots, B_k\} \subseteq A_1 \cup A_2$. The case that $k = 1$ is simple. When $k > 1$ it must be that for each $i \in [1, k], B_i \cap Var(\delta) \neq \emptyset$. The substitutions σ_1 and σ_2 assign to each variable $x \in S$ a term containing new variables $w_{l_1}, \ldots, w_{l_h}$ that correspond to those sets in $\{B_1, \ldots, B_k\}$ that contain x. These terms are such that the unification of $Eq(\sigma_1) \cup Eq(\sigma_2) \cup Eq(\delta)$ causes the unification of all these new variables. Therefore, in an mgu of $Eq(\sigma_1) \cup Eq(\sigma_2) \cup Eq(\delta)$ all the variables in S are bound to terms containing the same new variable and only that one. On the contrary, variables not in S will be bound to ground terms.

Theorem 6.4. *The abstract unification $\mathbf{U}_{Sh}$ is optimal with respect to $\mathbf{U}_{Rs}$.*

Proof. Let $[A_1, U_1], [A_2, U_2] \in$ `Sharing` with $U_1 \cap U_2 = \emptyset$. Let also $R_0 = A_1 \cup A_2$ and $U_0 = U_1 \cup U_2$ and $\delta = \{x_1/t_1, \ldots, x_n/t_n\} \in Subst$ such that $Var(\delta) \subseteq U_0$. The claim is:

$$\mathbf{U}_{Sh}([A_1, U_1], [A_2, U_2], \delta) = \alpha_{Rs,Sh}(\mathbf{U}_{Rs}(\gamma_{Sh,Rs}([A_1, U_1]), \gamma_{Sh,Rs}([A_2, U_2], \delta)).$$

In the light of the correctness shown in Theorem 6.3, we only need to show the following point **(A)**:

$$\textbf{(A)} \qquad \mathbf{U}_{Sh}([A_1, U_1], [A_2, U_2], \delta) \sqsubseteq_{Sh} \alpha_{Rs,Sh}(\mathbf{U}_{Rs}(\gamma_{Sh,Rs}([A_1, U_1]), \gamma_{Sh,Rs}([A_2, U_2], \delta))$$

Again, the notation introduced in Section 6.2 is used below. Let also for $i \in [1, 2]$, $[\Sigma_i, U_i] = \gamma_{Sh,Rs}([A_i, U_i])$. We will show the following statement **(X)** that immediately implies **(A)**:

(X) *$\forall S \in R_n$, there are $\sigma_1, \sigma_2 \in Subst$, such that:*
(a) $\sigma_1 \in \Sigma_1$, $\sigma_2 \in \Sigma_2$, and $Var(\sigma_1) \cap Var(\sigma_2) = \emptyset$;
(b) $\mathbf{u}_{Rs}(\sigma_1, \sigma_2, \delta)$ is successful;
(c) if $\mathbf{u}_{Rs}(\sigma_1, \sigma_2, \delta) = \rho_n$, *then* $\alpha_{Rs,Sh}([\{\rho_n\}, U_0]) = [\{S\}, U_0]$. [2]

[2] Observe that, for the sake of simplicity, we do not show the empty set in the first component of $[\{S\}, U_0]$. This causes no problem and is done throughout this proof.

Recall from the notation of Section 6.2 that $S = \bigcup K$, with $K = \{B_1, \ldots, B_k\} \subseteq R_0$. The following additional notation is needed:
- for each $B_i \in K$, let w_i be a distinct extra variable not in U_0;
- for each $x \in S$, $B_x = \{B \mid B \in K, x \in B\}$; notice that, by the hypothesis that $U_1 \cap U_2 = \emptyset$, if $x \in U_i$ then $\bigcup B_x \subseteq U_i$; in fact, we can partition K in two parts, $K_1 = \{B_i \in K \mid B_i \subseteq U_1\}$ and K_2 defined analogously; notice also that for any $x, y \in U_i$ we may have $B_x \cap B_y \neq \emptyset$;
- $N = max\{|B_x| \mid x \in S\}$;
- fixing $x \in S$, and letting $B_x = \{B_{l_1}, \ldots, B_{l_h}\}$, then the new variables associated to these sets will be denoted, $w_{l_1}, \ldots, w_{l_h}$.

The substitutions σ_1 and σ_2 will be defined through one substitution ρ_0 such that $supp(\rho_0) = U_0$; σ_1 and σ_2 can be recovered from ρ_0 by separating the bindings concerning U_1 from those concerning U_2; $var\text{-}range(\rho_0)$ consists only of the extra variables w_{l_j}. The substitution ρ_0 is defined in the following four points:
1. For each $x \in S \cap Var(\delta)$ there are two cases to distinguish:
   (a) If $x \in$ $var\text{-}range$ (δ), then

$$\rho_0(x) = s\,(\underbrace{c(w_{l_1}, w_{l_1}), c(w_{l_2}, w_{l_2}), \ldots, c(w_{l_h}, w_{l_h})}_{l_h \text{ times}}, \underbrace{c(w_{l_1}, w_{l_1}), \ldots, c(w_{l_1}, w_{l_1})}_{N - l_h \text{ times}})$$

   (b) If $x \in supp(\delta)$, then $x \equiv x_r$ for some $r \in [1, n]$, i.e., x is the left-hand side of the r-th binding, x_r/t_r, of δ. In this case, $\rho_0(x) = \zeta(t_r)$, where for each $y \in Var(t_r) \setminus S, \zeta(y) = a$, for some constant a, whereas if $y \in S \cap Var(t_r), \zeta(y)$ is as follows:

$$\zeta(y) = s\,(\underbrace{c(w_{l_1}, w_{l_2}), c(w_{l_2}, w_{l_3}), \ldots, c(w_{l_h}, w_{l_1})}_{l_h \text{ times}}, \underbrace{c(w_{l_1}, w_{l_1}), \ldots, c(w_{l_1}, w_{l_1})}_{N - l_h \text{ times}})$$

2. For each $x \in S \setminus Var(\delta), \rho_0(x) = s(w_{l_1}, \ldots, w_{l_h})$.
3. For each $x \in Var(\delta) \setminus S$ there are two cases to consider:
   (a) if $x \in var\text{-}range(\delta)$, then $\rho_0(x) = a$
   (b) if $x \in supp(\delta)$, then $x \equiv x_r$ for some binding x_r/t_r of δ. In this case, $\rho_0(x) = \zeta_a(t_r)$, where ζ_a binds to the constant a all variables in t_r.
4. For each $x \in U_0 \setminus (S \cup Var(\delta))$, $\rho_0(x) = a$, for some constant a.

Now, it remains to verify that ρ_0 satisfies statement **(X)**, i.e. that the bindings of ρ_0 can be partitioned into two substitutions $\sigma_1 \in \Sigma_1$ and $\sigma_2 \in \Sigma_2$ satisfying points (a) to (c).

Proof of point $(X.a)$**.** Let $\sigma_1 = \{x/t \in \rho_0 \mid x \in U_1\}$; σ_2 is defined similarly. Obviously, these two substitutions have disjoint *supp* sets. They have also disjoint *var-range* sets because for each x in U_1, $Var(\rho_0(x)) \subseteq \{w_{l_1}, \ldots, w_{l_h}\}$ and each of these variables corresponds to a set in $B_x \subseteq A_1$. Thus, for any $y \in U_2$, $Var(\rho_0(y)) \cap Var(\rho_0(x)) = \emptyset$. To see that $\sigma_i \in \Sigma_i$, it suffices to observe that the construction of ρ_0 yields the following point (Y):

(Y) $\bigcup K_i \subseteq$ supp (σ_i), *and the variables in* $\bigcup K_i$ *are exactly the variables not ground in* σ_i, *and for each such variable* x, $Var(\sigma_i(x)) = \{w_{l_1}, \ldots, w_{l_h}\}$.

From point (Y) it follows immediately that $\alpha_{Rs\,Sh}([\{\sigma_i\}, U_i]) = [K_i, U_i] \sqsubseteq [A_i, U_i]$, and thus $\sigma_i \in \Sigma_i$.

Proof of point (X.b). Consider for each equation $(x_i = t_i) \in Eq(\delta)$, the two terms $\rho_0(x_i)$ and $\rho_0(t_i)$. In order to show that they are unifiable, we distiguish two cases:

1. One of them is ground: this may arise either when $x_i \notin S$ or when $Var(t_i) \cap S = \emptyset$.
   - When $x_i \notin S$ (i.e., case 3(b) of the def. of ρ_0), by Fact 1 of Section 6.3, we have $Var(t_i) \cap S = \emptyset$. Thus, $\rho_0(t_i)$ is ground too, and, by definition of ρ_0, it is identical to $\rho_0(x_i)$.
   - The case $Var(t_i) \cap S = \emptyset$ is analogous to the previous one.
2. Both terms are non-ground terms. In this case, the following points come easily from the definition of ρ_0 above:

   ($\mathbf{Z}_1$) the only variables in the equation $\rho_0(x_i) = \rho_0(t_i)$ are extra variables w_i;

   ($\mathbf{Z}_2$) $\rho_0(x_i)$ and $\rho_0(t_i)$ differ only for having different variables in corresponding leaf positions;

   ($\mathbf{Z}_3$) solving the equation $\rho_0(x_i = t_i)$ causes only the unification of all the present variables with each other, leaving all variables free.

   Thus, in both cases, the unification is successful.

Proof of point (X.c). Two cases are distinguished:

1. Case $S \cap Var(\delta) = \emptyset$. Here, $K = \{S\}$ and $S \in A_i$ for some $i \in [1, 2]$. Let us fix such an i. By the proof of point (X.a), $\alpha_{RsSh}([\{\rho_0\}, U_i]) = [\{S\}, U_i]$. From the same proof, it follows also that in ρ_0 only the variables in S are not ground, and that they all share only the (extra) variable $w \notin U_0$ corresponding to S. Hence, $\alpha_{RsSh}([\{\rho_0\}, U_0]) = [\{S\}, U_0]$. Now, observe that, by point (X.b)(1), the equations $\rho_0(x_i = t_i)$ are ground identities. This means that they are simply eliminated by the unification algorithm. Thus, $\rho_n = \rho_0$, as expected.
2. Case $S \cap Var(\delta) \neq \emptyset$. Let us examine the computation of $mgu(Eq(\rho_0) \cup Eq(\delta))$. It is well-known that, in computing an (idempotent) *mgu* of a system of equations, we may consider the equations in any order. We consider the unification of $\rho_0(Eq(\delta))$ first, and then we apply the computed *mgu* to $Eq(\rho_0)$.
   - Point (Z_1) above, shows that only extra variables occur in $\rho_0(Eq(\delta))$;
   - Fact 2 of Section 6.3 implies that all the extra variables $w_1, \ldots, w_k$ occur in $\rho_0(Eq(\delta))$;
   - Fact 3 together with point (Z_3) prove that, in the unification process, all the (extra) variables present in $\rho_0(Eq(\delta))$ are unified in a single variable. Thus an *mgu* of $\rho_0(Eq(\delta))$ is $\mu = \{w_1/w_k, \ldots, w_{k-1}/w_k\}$. We can substitute $Eq(\mu)$ to $\rho_0(Eq(\delta))$ and continue the unification with $Eq(\mu) \cup Eq(\rho_0)$;
   - Point (Y) guarantees that extra-variables w_i may appear only in the var-range of ρ_0. Hence, by the definition of μ, $\mu(Eq(\rho_0))$ is in solved form. It is also easy to see that $Eq(\mu) \cup \mu(Eq(\rho_0))$ is in solved form. Thus, it represents an idempotent *mgu* of our initial system $Eq(\rho_0) \cup Eq(\delta)$. The corresponding *mgu* is $\rho_n = \mu \circ \rho_0$. From point (Y) and the definition of μ it follows that in ρ_n exactly the variables of S are not ground and that they all share the extra variable w_k, that is the only variable in *var-range* (ρ_n).

From the previous points, we get $\alpha_{RsSh}([\{\rho_n\}, U_0]) = [\{S\}, U_0]$ (recall that $w_k \notin U_0$, and thus it is filtered out by the abstraction). This completes the proof of the the desired statement (X) and of the whole Theorem as well. □

As a final remark, observe that $\mathbf{U}_{Sh}$ is not complete. For instance, consider $U_1 = \{x\}$, $U_2 = \{z\}$, and let $\sigma = \{x/f(a,w)\}$, and $\delta = \{x/f(z,a)\}$. In this case, $\alpha_{RsSh}(\mathbf{U}_{Rs}([\{\sigma_1\}, U_1], [\{\emptyset\}, U_2], \delta))$ is equal to $[\{\emptyset\}, U_1 \cup U_2]$, whereas the corresponding abstract unification $\mathbf{U}_{Sh}([\{\{x\},\emptyset\}, U_1], [\{\{z\},\emptyset\}, U_2], \delta)$ results to be equal to $[\{\{x,z\},\emptyset\}, U_1 \cup U_2]$.

7. Conclusions and related works

Recently, several researchers have studied the domain `Sharing` and its operations. Codish et al. in Ref. [10] propose an alternative way of representing set-sharing information. In place of collecting the sets of variables that may share a common variable, as `Sharing` does, they associate to each variable the set of variables that may be contained in the terms bound to that variable. Clearly this new domain is isomorphic to `Sharing`. In Ref. [10] the abstract operations of this new domain are also discussed. The abstract unification is based on an associative commutative and idempotent equality theory. Abstract *lub* is shown to be complete, whereas only the correctness of the abstract unification and projection are shown (cf. Theorem 7 of Ref. [10]). In the light of the optimality result shown in the present paper, it is expectable that also the abstract unification of Ref. [10] can be proven optimal.

In Ref. [9] it is proved that a strict abstraction, SS^ρ, of `Sharing` is sufficient for computing pair-sharing. This means that, when analyzing programs, in place of using `Sharing`, we may use the more abstract SS^ρ obtaining exactly the same pair-sharing information. This is important because abstract unification in SS^ρ is polynomial whereas that in `Sharing` is exponential.

It is worth mentioning that, as Bagnara et al. remark in Ref. [9], SS^ρ is the quotient of `Sharing` with respect to pair-sharing, where the notion of quotient was introduced in Ref. [7]. [3]

In the light of the result of Bagnara et al., it is natural to wonder whether there are analyses for which the `Sharing` domain is still needed or for all practical applications it can be replaced by the more efficient SS^ρ.

In order to answer this question, we need to explain briefly the relation between SS^ρ and `Sharing`. For each element $[S, U] \in$ `Sharing`, the corresponding abstraction in SS^ρ is $[\rho(S), U]$, where, intuitively, $\rho(S)$ adds to S every subset of U that does not introduce any new pair-sharing to that already expressed by S.

Thus, for instance, SS^ρ cannot distinguish between the following two values of `Sharing`: $S_1 = [\{\{x,y\},\{x,z\},\{y,z\}\}, U]$ and $S_2 = [\{\{x,y\},\{x,z\},\{y,z\}, \{x,y,z\}\}, U]$, where $U = \{x,y,z\}$. Observe in fact that:

$$\begin{aligned}\rho(\{\{x,y\},\{x,z\},\{y,z\}\}) &= \rho(\{\{x,y\},\{x,z\},\{y,z\},\{x,y,z\}\}) \\ &= \{\{x,y\},\{x,z\},\{y,z\},\{x,y,z\}\}\end{aligned}$$

Being able to distinguish between these two sharing situations may be important for optimizations concerning the *AND* parallel execution of Prolog programs. Assume we want to execute $p(x), q(y), r(z)$ by means of parallel processes. In situations rep-

[3] Observe that Sharing is also studied in [7], where its quotient with respect to groundness information is characterized as the well-known domain `Def` consisting of definite propositional formulas.

resented by S_1 we are sure that no variable is shared by more than two processes, whereas in the situations represented by S_2 all three processes may share a variable. Good strategies for deciding which goals should be solved in parallel might use such information.

Finally, it is interesting to observe that the relation between `Sharing` and pair-sharing has also been studied in Ref. [15], with a different goal: to use the complementation operation Ref. [8], in order to decompose the domain `Sharing` into three simpler domains, each expressing one of the different information that coexist in `Sharing`, namely, groundness, pair-sharing and set-sharing information. The reduced product of these three components gives `Sharing` back. It is worthwhile to mention that in that paper it is shown that `Sharing` cannot be obtained as the reduced product of a domain *PS* expressing only pair-sharing with some other domain more abstract than `Sharing`. More precisely, in Ref. [15] it is shown that the complement of *PS* with respect to `Sharing` is `Sharing` itself.

Acknowledgements

The work has been partially supported by the Italian MURST project 9701248444-044. Thanks to the anonymous referees for in-depth reading and helpful comments.

References

[1] P. Cousot, R. Cousot, Abstract interpretation: A unified framework for static analysis of programs by construction of approximation of fixpoints, Proceedings of the 4th ACM Symposium on Principles of Programming Languages and Systems, ACM, New York, 1977, pp. 238–252.

[2] K. Mutukumar, M. Hermenegildo, Combined determination of sharing and freeness of program variables through abstract interpretation, in: Proceedings of 8th International Conference on Logic Programming ICLP91, Paris, MIT Press, Cambridge, MA, 1991, pp. 49–63.

[3] M. Codish, D. Dams, G. Filè, M. Bruynhooghe, On the design of a correct freeness analysis for logic programs, The Journal of Logic Programming 28 (3) (1996) 181–206.

[4] A. Bossi, N. Cocco, Programs without failures, in: N. Fuchs (Ed.), Proceedings of the 7th Logic Programming Synthesis and Transformation LOPSTR97, Leuven, Belgium, LNCS, Springer, Berlin, to appear.

[5] S. Debray, P. Lopez Garcia, M. Hermenegildo, Non-Failure analysis for logic programs, Proceedings of the 14th International Conference on Logic Programming, MIT Press, Cambridge, MA, 1997, pp. 48–62.

[6] D. Jacobs, A. Langen, Accurate and efficient approximation of variable aliasing in logic programs, The Journal of Logic Programming 13 (1992) 291–314.

[7] A. Cortesi, G. Filè, W. Winsborough, The quotient of an abstract interpretation, Theoretical Computer Science 202 (1-2) (1998) 163–192.

[8] A. Cortesi, G. Filè, R. Giacobazzi, C. Palamidessi, F. Ranzato, Complementation in abstract interpretation, ACM Transactions on Programming Languages and Systems 19 (1) (1997) 7–47.

[9] R. Bagnara, P. Hill, E. Zaffanella, Set-sharing is redundant for pair-sharing, in: P. Van Hentenryck (Ed.), Static Analysis: Proceedings of the 4th International Symposium, Paris, France, LNCS vol. 1302, Springer, Berlin, 1997, pp. 53–67.

[10] M. Codish, V. Lagoon, F. Bueno, An algebraic approach to sharing analysis of logic programs, in: P. Van Hentenryck (Ed.), Static Analysis: Proceedings of the 4th International Symposium, Paris, France, LNCS vol. 1302, Springer, Berlin, 1997, pp. 68–82.

[11] A. Mycroft, Completeness and predicate-based abstract interpretation, in: Proceedings of the ACM Symposium on Partial Evaluation and Program Manipulation, PEPM93, ACM, New York, 1993, pp. 179–185.

[12] J-L. Lassez, M. Maher, K. Marriott, Unification revisited, in: Jack Minker (Ed.), Foundations of Deductive Databases and Logic Programming, Morgan-Kaufmanm, Los Altos, CA, 1987, pp. 587–625.

[13] P. Cousot, R. Cousot, Abstract interpretation and applications to logic programs, The Journal of Logic Programming 13 (1–4) (1992) 103–179.

[14] A. Cortesi, G. Filè, W. Winsborough, Optimal groundness analysis using propositional logic, The Journal of Logic Programming 27 (2) (1996) 137–167.

[15] G. Filè, F. Ranzato F, Complementation of abstract domains made easy, in: M. Maher (Ed.), Proceedings of the Joint International Conference and Symposium on Logic Programming JICSLP96, MIT Press, Cambridge, MA, 1996, pp. 348–362.